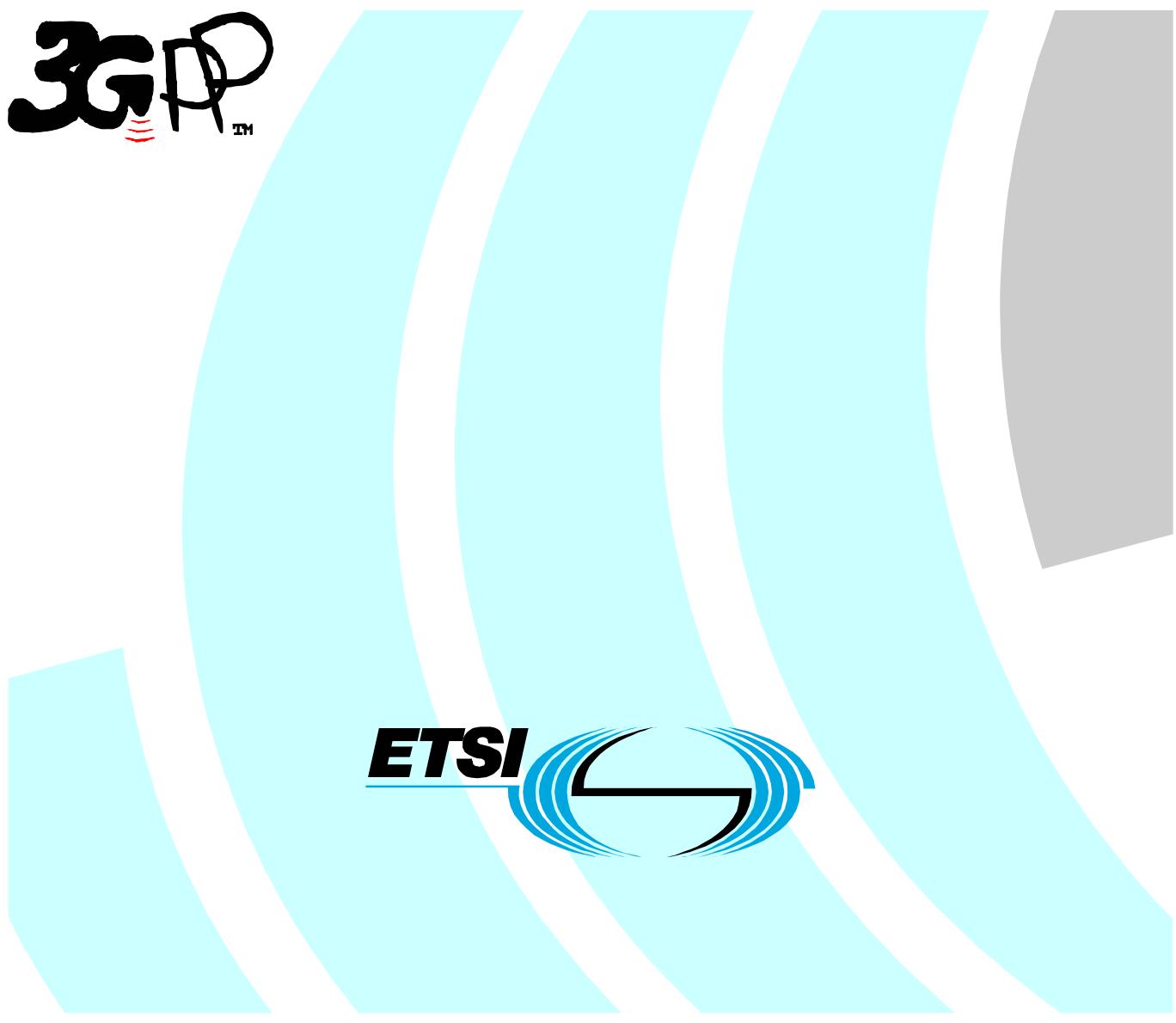


**Universal Mobile Telecommunications System (UMTS);  
Open Service Access (OSA)  
Application Programming Interface (API);  
Part 4: Call control;  
Subpart 4: Multimedia call control data SCF  
(3GPP TS 29.198-04-4 version 5.1.0 Release 5)**



---

Reference

RTS/TSGN-0529198-04-4v510

---

Keywords

UMTS

***ETSI***

---

650 Route des Lucioles  
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C  
Association à but non lucratif enregistrée à la  
Sous-Préfecture de Grasse (06) N° 7803/88

---

***Important notice***

Individual copies of the present document can be downloaded from:  
<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status.  
Information on the current status of this and other ETSI documents is available at  
<http://portal.etsi.org/tb/status/status.asp>

If you find errors in the present document, send your comment to:  
[editor@etsi.fr](mailto:editor@etsi.fr)

---

***Copyright Notification***

No part may be reproduced except as authorized by written permission.  
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2002.  
All rights reserved.

**DECT™, PLUGTESTS™ and UMTS™** are Trade Marks of ETSI registered for the benefit of its Members.  
**TIPHON™** and the **TIPHON logo** are Trade Marks currently being registered by ETSI for the benefit of its Members.  
**3GPP™** is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.

---

## Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: "*Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards*", which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (<http://webapp.etsi.org/IPR/home.asp>).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

---

## Foreword

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under [www.etsi.org/key](http://www.etsi.org/key).

---

# Contents

Intellectual Property Rights .....	2
Foreword.....	2
Foreword.....	5
Introduction .....	5
1    Scope .....	7
2    References .....	7
3    Definitions and abbreviations.....	8
3.1    Definitions.....	8
3.2    Abbreviations .....	8
4    MultiMedia Call Control Service Sequence Diagrams .....	8
4.1    Barring for media combined with call routing, alternative 1 .....	8
4.2    Barring for media combined with call routing, alternative 2.....	10
4.3    Barring for media, simple.....	11
4.4    Call Volume charging supervision .....	12
5    Class Diagrams.....	14
6    MultiMedia Call Control Service Interface Classes .....	15
6.1    Interface Class IpMultiMediaCallControlManager .....	16
6.1.1    Method createMediaNotification() .....	16
6.1.2    Method destroyMediaNotification() .....	17
6.1.3    Method changeMediaNotification().....	17
6.1.4    Method getMediaNotification() .....	18
6.2    Interface Class IpAppMultiMediaCallControlManager .....	18
6.2.1    Method reportMediaNotification() .....	18
6.3    Interface Class IpMultiMediaCall .....	19
6.3.1    Method superviseVolumeReq() .....	19
6.4    Interface Class IpAppMultiMediaCall .....	20
6.4.1    Method superviseVolumeRes() .....	20
6.4.2    Method superviseVolumeErr() .....	21
6.5    Interface Class IpMultiMediaCallLeg .....	21
6.5.1    Method mediaStreamAllow() .....	21
6.5.2    Method mediaStreamMonitorReq()	22
6.5.3    Method getMediaStreams() .....	22
6.6    Interface Class IpAppMultiMediaCallLeg .....	22
6.6.1    Method mediaStreamMonitorRes() .....	23
6.7    Interface Class IpMultiMediaStream.....	23
6.7.1    Method subtract().....	23
7    MultiMedia Call Control Service State Transition Diagrams .....	24
8    Multi-Media Call Control Data Definitions .....	25
8.1    Event Notification Data Definitions .....	25
8.1.1    TpMediaStreamRequestSet .....	25
8.1.2    TpMediaStreamRequest.....	25
8.1.3    TpMediaStreamDirection .....	25
8.1.4    TpMediaStreamDataRequest.....	26
8.1.5    TpAudioCapabilitiesType .....	26
8.1.6    TpVideoCapabilitiesType .....	26
8.1.7    TpDataCapabilities .....	27
8.1.8    TpMediaStreamEventType .....	27
8.1.9    TpMediaStreamSet .....	27
8.1.10    TpMediaStream .....	27
8.1.11    TpMediaStreamDataType.....	27

8.2	Multi-Media Call Control Data Definitions .....	27
8.2.1	IpMultiMediaCall .....	27
8.2.2	IpMultiMediaCallRef.....	27
8.2.3	IpAppMultiMediaCall ..	27
8.2.4	IpAppMultiMediaCallRef.....	27
8.2.5	IpMultiMediaCallLeg .....	28
8.2.6	IpMultiMediaCallLegRef .....	28
8.2.7	IpAppMultiMediaCallLeg .....	28
8.2.8	IpAppMultiMediaCallLegRef.....	28
8.2.9	TpAppMultiMediaCallLegRefSet .....	28
8.2.10	TpMultiMediaCallIdentifier .....	28
8.2.11	TpMultiMediaCallIdentifierSet .....	28
8.2.12	TpMultiMediaCallLegIdentifier .....	28
8.2.13	IpAppMultiMediaCallControlManager .....	28
8.2.14	IpAppMultiMediaCallControlManagerRef .....	28
8.2.15	TpAppMultiMediaCallBack .....	29
8.2.16	TpAppMultiMediaCallBackRefType .....	29
8.2.17	TpAppMultiMediaCallLegCallBack .....	29
8.2.18	TpCallSuperviseVolume.....	29
8.2.19	TpNotificationMediaRequest.....	30
8.2.20	TpMediaNotificationRequested.....	30
8.2.21	TpMediaNotificationsRequestedSet .....	30
<b>Annex A (normative):</b>	<b>OMG IDL Description of Multi-Media Call Control SCF.....</b>	<b>31</b>
<b>Annex B (informative):</b>	<b>W3C WSDL Description of Multi-Media Call Control SCF .....</b>	<b>32</b>
<b>Annex C (informative):</b>	<b>Java API Description of the Call Control SCFs.....</b>	<b>33</b>
<b>Annex D (informative):</b>	<b>Change history .....</b>	<b>34</b>
History .....		36

---

## Foreword

This Technical Specification has been produced by the 3<sup>rd</sup> Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

---

## Introduction

The present document is part 4, sub-part 4 of a multi-part TS covering the 3<sup>rd</sup> Generation Partnership Project: Technical Specification Group Core Network; Open Service Access (OSA); Application Programming Interface (API), as identified below. The **API specification** (3GPP TS 29.198) is structured in the following Parts:

Part 1:	"Overview";	
Part 2:	"Common Data Definitions";	
Part 3:	"Framework";	
<b>Part 4:</b>	<b>"Call Control";</b>	
	Sub-part 1: "Call Control Common Definitions";	(new in 3GPP Release 5)
	Sub-part 2: "Generic Call Control SCF";	(new in 3GPP Release 5)
	Sub-part 3: "Multi-Party Call Control SCF";	(new in 3GPP Release 5)
	<b>Sub-part 4: "Multi-Media Call Control SCF";</b>	<b>(new in 3GPP Release 5)</b>
	Sub-part 5: "Conference Call Control SCF";	(not part of 3GPP Release 5)
Part 5:	"User Interaction SCF";	
Part 6:	"Mobility SCF";	
Part 7:	"Terminal Capabilities SCF";	
Part 8:	"Data Session Control SCF";	
Part 9:	"Generic Messaging SCF";	(not part of 3GPP Release 5)
Part 10:	"Connectivity Manager SCF";	(not part of 3GPP Release 5)
Part 11:	"Account Management SCF";	
Part 12:	"Charging SCF".	
Part 13:	"Policy Management SCF";	(new in 3GPP Release 5)
Part 14:	"Presence and Availability Management SCF";	(new in 3GPP Release 5)

The **Mapping specification of the OSA APIs and network protocols** (3GPP TR 29.998) is also structured as above. A mapping to network protocols is however not applicable for all Parts, but the numbering of Parts is kept. Also in case a Part is not supported in a Release, the numbering of the parts is maintained.

**Table: Overview of the OSA APIs & Protocol Mappings 29.198 & 29.998-family**

<b>OSA API specifications 29.198-family</b>					<b>OSA API Mapping - 29.998-family</b>		
29.198-01	Overview				29.998-01	Overview	
29.198-02	Common Data Definitions				29.998-02	<i>Not Applicable</i>	
29.198-03	Framework				29.998-03	<i>Not Applicable</i>	
<b>Call Control (CC) SCF</b>	29.198-04-1	29.198-04-2	29.198-04-3	<b>29.198-04-4 Multi-media CC SCF</b>	29.998-04-1	Generic Call Control – CAP mapping	
	Common CC data definitions	Generic CC SCF	Multi-Party CC SCF		29.998-04-2	<i>Generic Call Control – INAP mapping</i>	
					29.998-04-3	<i>Generic Call Control – Megaco mapping</i>	
					29.998-04-4	Multiparty Call Control – SIP mapping	
29.198-05	User Interaction SCF				29.998-05-1	User Interaction – CAP mapping	
					29.998-05-2	<i>User Interaction – INAP mapping</i>	
					29.998-05-3	<i>User Interaction – Megaco mapping</i>	
					29.998-05-4	User Interaction – SMS mapping	
29.198-06	Mobility SCF				29.998-06	User Status and User Location – MAP mapping	
29.198-07	Terminal Capabilities SCF				29.998-07	<i>Not Applicable</i>	
29.198-08	Data Session Control SCF				29.998-08	Data Session Control – CAP mapping	
29.198-09	<i>Generic Messaging SCF</i>				29.998-09	<i>Not Applicable</i>	
29.198-10	<i>Connectivity Manager SCF</i>				29.998-10	<i>Not Applicable</i>	
29.198-11	Account Management SCF				29.998-11	<i>Not Applicable</i>	
29.198-12	Charging SCF				29.998-12	<i>Not Applicable</i>	
29.198-13	Policy Management SCF				29.998-13	<i>Not Applicable</i>	
29.198-14	Presence & Availability Management SCF				29.998-14	<i>Not Applicable</i>	

The present document is equivalent to ETSI ES 202 915-04-4 v1.1.1 (Parlay 4.0).

## 1 Scope

The present document is Part 4, Sub-part 4 of the Stage 3 specification for an Application Programming Interface (API) for Open Service Access (OSA).

The OSA specifications define an architecture that enables application developers to make use of network functionality through an open standardised interface, i.e. the OSA APIs. The concepts and the functional architecture for the OSA are contained in 3GPP TS 23.127 [3]. The requirements for OSA are contained in 3GPP TS 22.127 [2].

The present document specifies the Multi-Media Call Control Service Capability Feature (SCF) aspects of the interface. All aspects of the Multi-Media Call Control SCF are defined here, these being:

- Sequence Diagrams
- Class Diagrams
- Interface specification plus detailed method descriptions
- State Transition diagrams
- Data definitions
- IDL Description of the interfaces
- WSDL Description of the interfaces
- Reference to the Java API description of the interfaces

The process by which this task is accomplished is through the use of object modelling techniques described by the Unified Modelling Language (UML).

This specification has been defined jointly between 3GPP TSG CN WG5, ETSI SPAN 12 and the Parlay Consortium, in co-operation with a number of JAIN™ Community member companies.

## 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

- [1] 3GPP TS 29.198-1 "Open Service Access; Application Programming Interface; Part 1: Overview".
- [2] 3GPP TS 22.127: "Stage 1 Service Requirement for the Open Service Access (OSA) (Release 5)".
- [3] 3GPP TS 23.127: "Virtual Home Environment (Release 5)".
- [4] 3GPP TS 22.002: "Circuit Bearer Services Supported by a PLMN".
- [5] ISO 4217 (1995): "Codes for the representation of currencies and funds".
- [6] 3GPP TS 24.002: "GSM-UMTS Public Land Mobile Network (PLMN) Access Reference Configuration".

[7] 3GPP TS 22.003: "Circuit Teleservices supported by a Public Land Mobile Network (PLMN)".

---

## 3 Definitions and abbreviations

### 3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 29.198-1 [1] apply.

### 3.2 Abbreviations

For the purposes of the present document, the abbreviations given in TS 29.198-1 [1] apply.

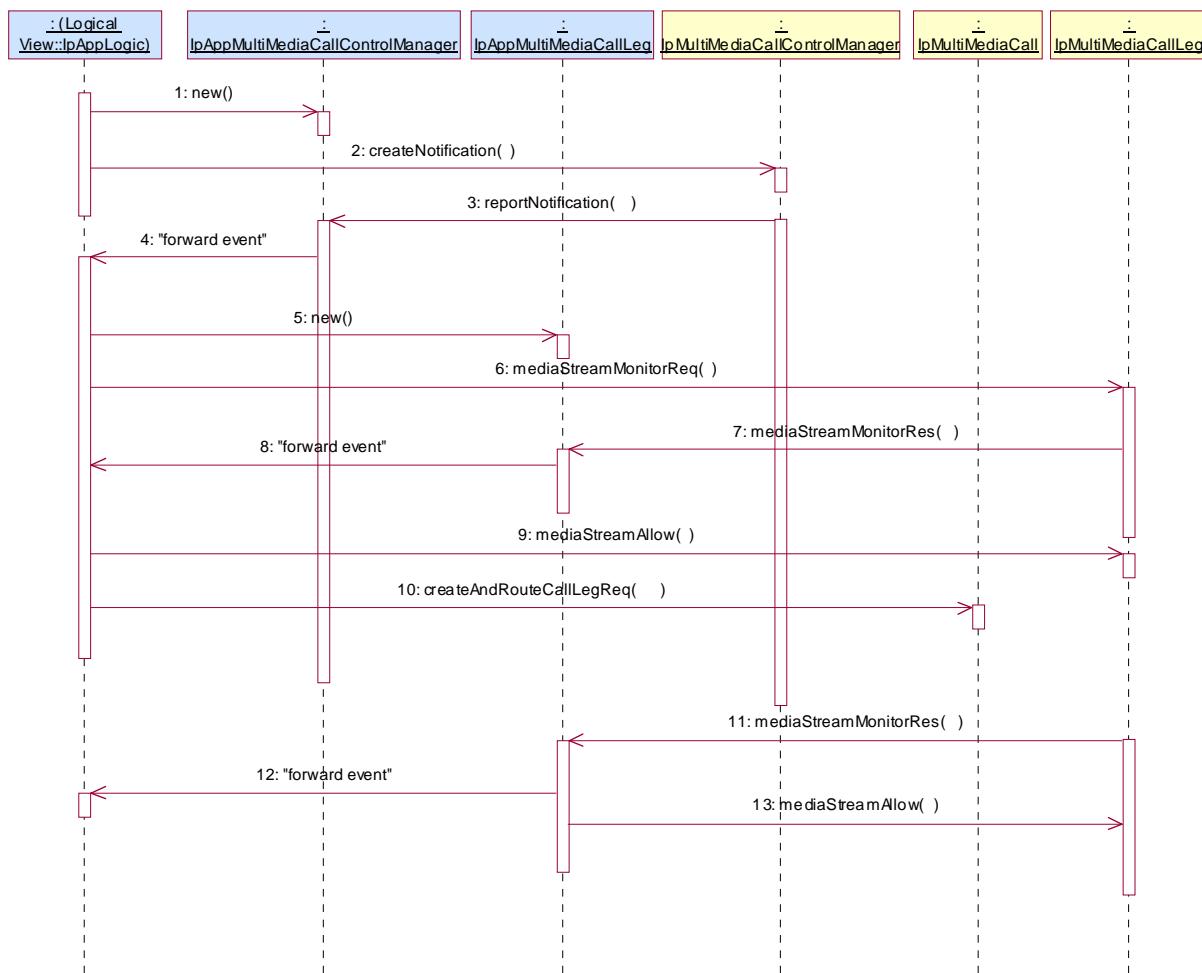
---

## 4 MultiMedia Call Control Service Sequence Diagrams

### 4.1 Barring for media combined with call routing, alternative 1

This sequence illustrates how one application can influence both the call routing and the media stream establishment of one call.

In this sequence there is one application handling both the media barring and the routing of the call.



- 1: The application creates a AppMultiMediaCallControlManager interface in order to handle callback methods.
- 2: The application expresses interest in all calls from subscriber A. Since createNotification is used and not createMediaNotification all calls are reported regardless of the media used.
- 3: A makes a call with the SIP INVITE with SDP media stream indicating video. The application is notified.
- 4: The event is forwarded to the application.
- 5: The application creates a new AppMultiMediaCallLeg interface to receive callbacks.
- 6: The application sets a monitor on video media streams to be established (added) for the indicated leg.
- 7: Since the video media stream was included in the SIP invite, the media streams monitored will be returned in the monitor result.
- 8: The event is forwarded to the application.
- 9: The application denies the video media stream, i.e., it is not included in the allowed media streams. This corresponds to removing the media stream from the setup.
- 10: The application requests to reroute the call to a different destination (or the same one...)
- 11: Later in the call the A party tries to establish a lower bandwidth video media stream. This is again reported with MediaStreamMonitorRes.
- 12: The event is forwarded.

13: This time the application allows the establishment of the media stream by including the media stream in the allowed list.

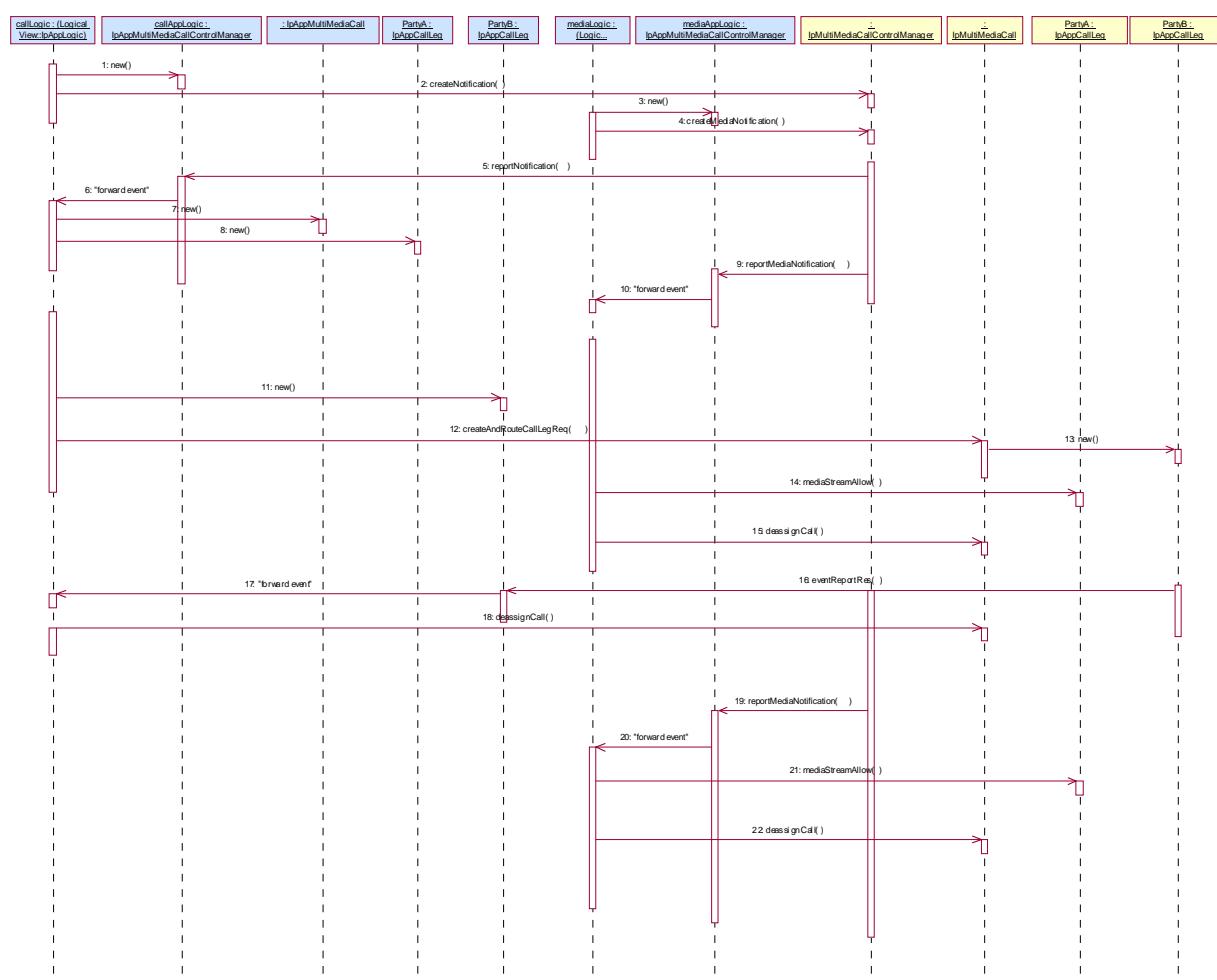
## 4.2 Barring for media combined with call routing, alternative 2

This sequence illustrates how one application can influence both the call routing and the media establishment of one call.

Media establishment and call establishment are regarded separately by the application.

From the gateway point of view it can actually be regarded as two separately triggered applications, one for media control and one for routing. This is also the way that it is shown here, for clarity.

However, an implementation of the application could combine the media logic and call logic in one object.



- 1: The application creates a new AppMultiMediaCallControlManager interface.
- 2: The application expresses interest in all calls from subscriber A for rerouting purposes.
- 3: The application creates a new AppMultiMediaCallControlManager interface. This is to be used for the media control only.
- 4: Separately the application expresses interest in some media streams for calls from and to A. The request indicates interrupt mode.

5: Subscriber A makes a call with the SIP INVITE with SDP media stream indicating video. Since the media establishment is combined with the SIP INVITE message, both applications are triggered (not necessarily in the order shown).

Here the call application is notified about the call setup.

6: The event is forwarded to the call control application.

7: The call control application creates a new AppMultiMediaCall interface.

8: The call control application creates a new AppMultiMediaCallLeg interface.

9: The media application is notified about the call setup. All media streams from the setup will be indicated.

10: The event is forwarded to the media application.

11: The call control application creates a new AppMultiMediaCallLeg interface.

12: The call application decides to reroute the call to another address. Included in the request are monitors on answer and call end.

However, since the media was also triggered in mode interrupt the call will not proceed until the media streams are confirmed or rejected.

14: The application allows the audio media stream, but refuses the high bandwidth video, by excluding it from the allowed list. Since both call processing and media handling is now acknowledged, the call routing can continue (with a changed SDP parameter reflecting the manipulated media).

15: The Media application is no longer interested in the call.

16: When the B subscriber answers the call application is notified.

17: The event is forwarded to the call application.

19: When later in the call A tries to establish a lower bandwidth video stream the media application is triggered.

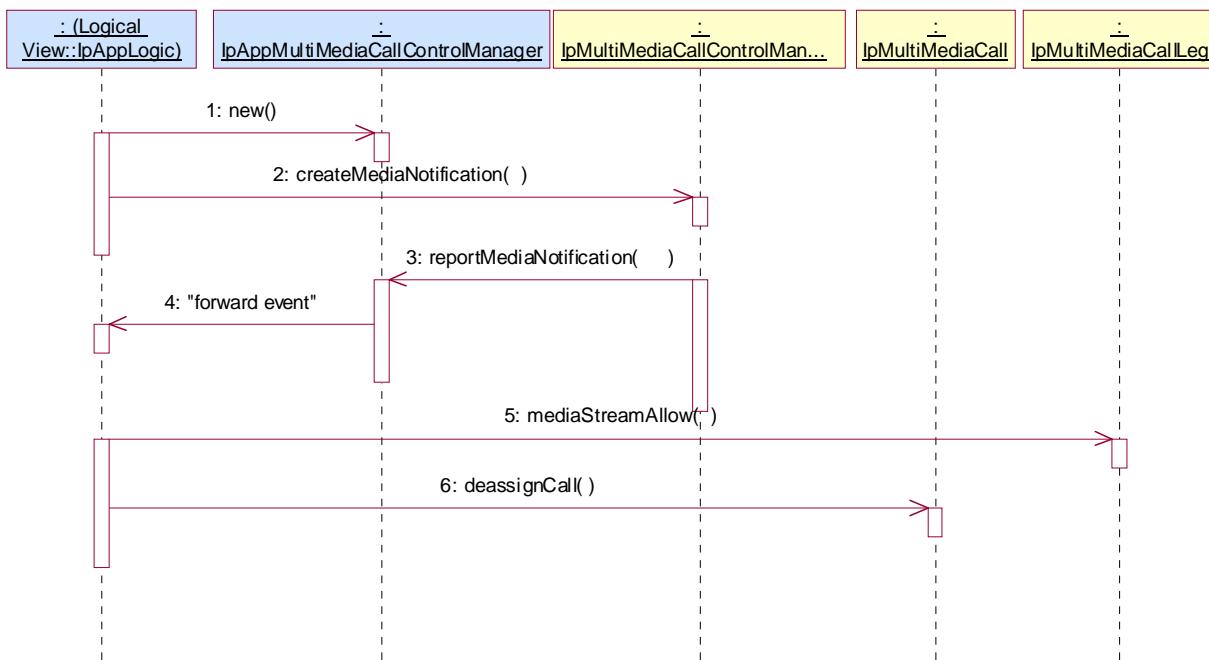
20: The triggering is forwarded to the media application.

21: The application now allows the establishment of the media stream by including the media stream in the mediaStreamAllow list.

22: The media application is no longer interested in the call.

### 4.3 Barring for media, simple

This sequence illustrates how an application can block the establishment of video streams for a certain user.

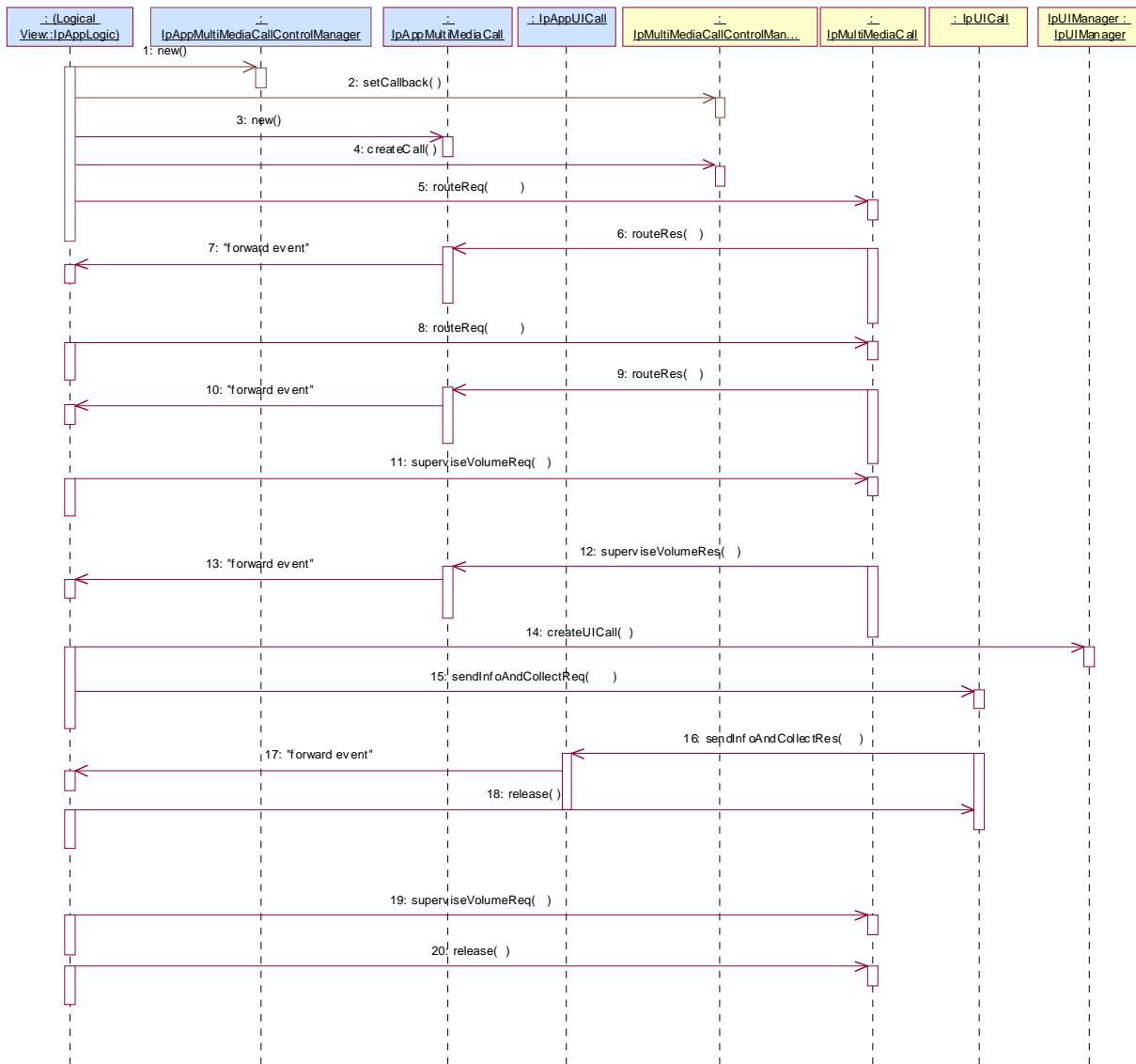


- 1: The application starts a new AppMultiMediaCallControlManager interface for reception of callbacks.
- 2: The application expresses interest in all calls from or to subscriber A that use video. The just created App interface is given as the callback interface.
- 3: Subscriber A makes a call with the SIP INVITE with SDP media stream indicating video.
- 4: The message is forwarded to the application.
- 5: The application indicates that the setup of the media stream is not allowed by not including the media stream in the allowed list. This has the effect of suppressing the video capabilities in the setup.
- 6: The application is no longer interested in the call.

New attempts to open video streams will again be indicated with a createMediaNotification.

## 4.4 Call Volume charging supervision

This sequence illustrates how an application may supervise a call based on the number of bytes that are exchanged.



- 1: The application creates a new interface to receive callbacks on the call control manager.
- 2: The created interface is set as the callback interface for the call control manager.
- 3: The application creates a new interface to receive callback on the call.
- 4: The application requests the creation of a call.
- 5: The application initiates the call by routing to the origination. This will implicitly create a call leg. The application requests a notification when the party answers.
- 6: When the A party answers the application is notified.
- 7: The message is forwarded to the logic.
- 8: The application also routes the call to the destination. This implicitly creates a call leg. The application requests to be notified on answer of the B-party.
- 9: When the B-party answers the application is notified.
- 10: The message is forwarded to the logic.

11: The application requests to supervise the call. In the request the application specifies a limit on the amount of bytes that may be transferred. The application specifies that if the limit is reached the application should be notified.

12: When the limit is reached a notification is send to the application.

13: The message is forwarded to the logic.

15: The application plays an announcement to the user, asking whether the user wants to end the call or continue the call.

16: When the user answers whether the call should continue.

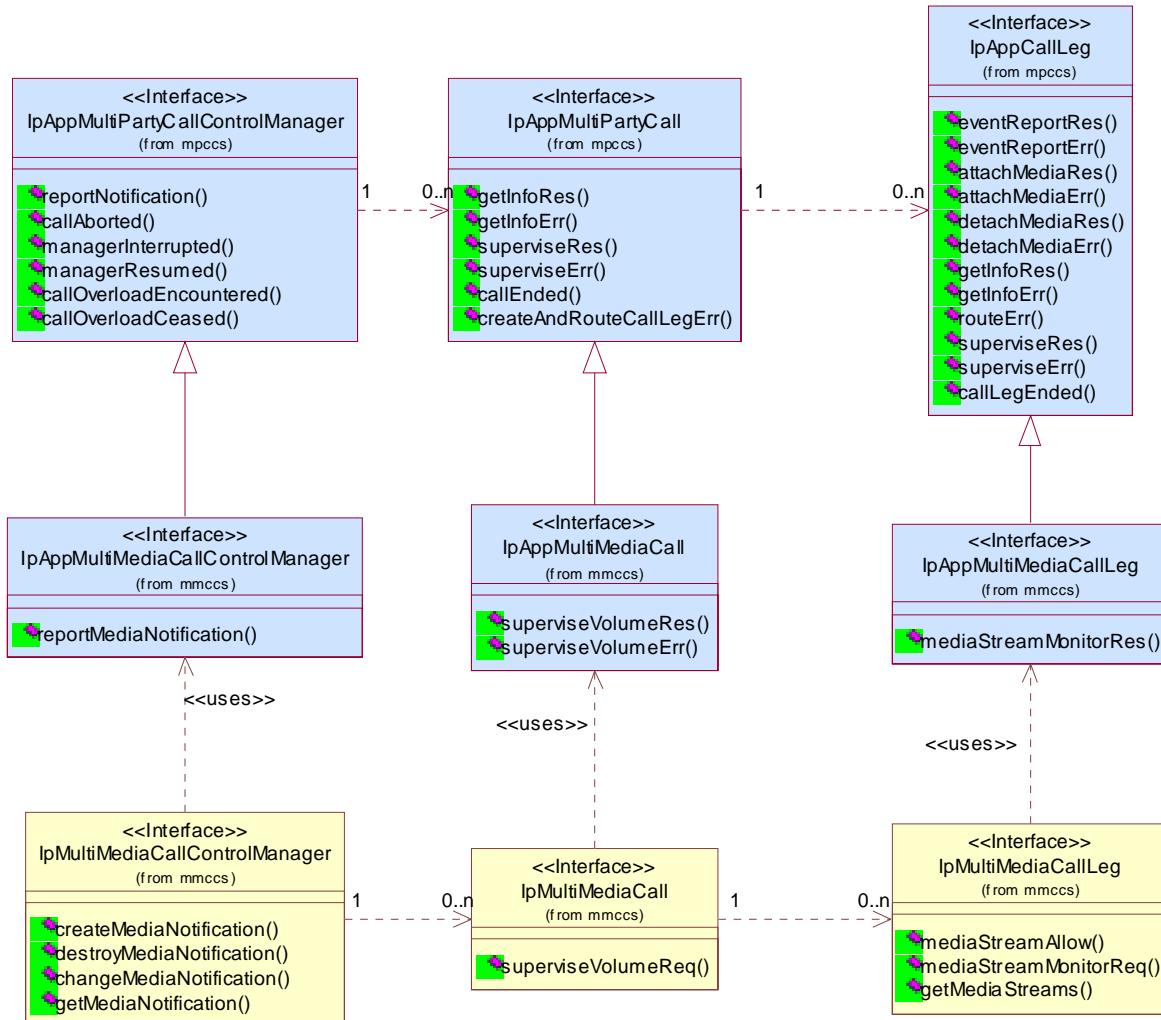
17: The message is forwarded to the logic.

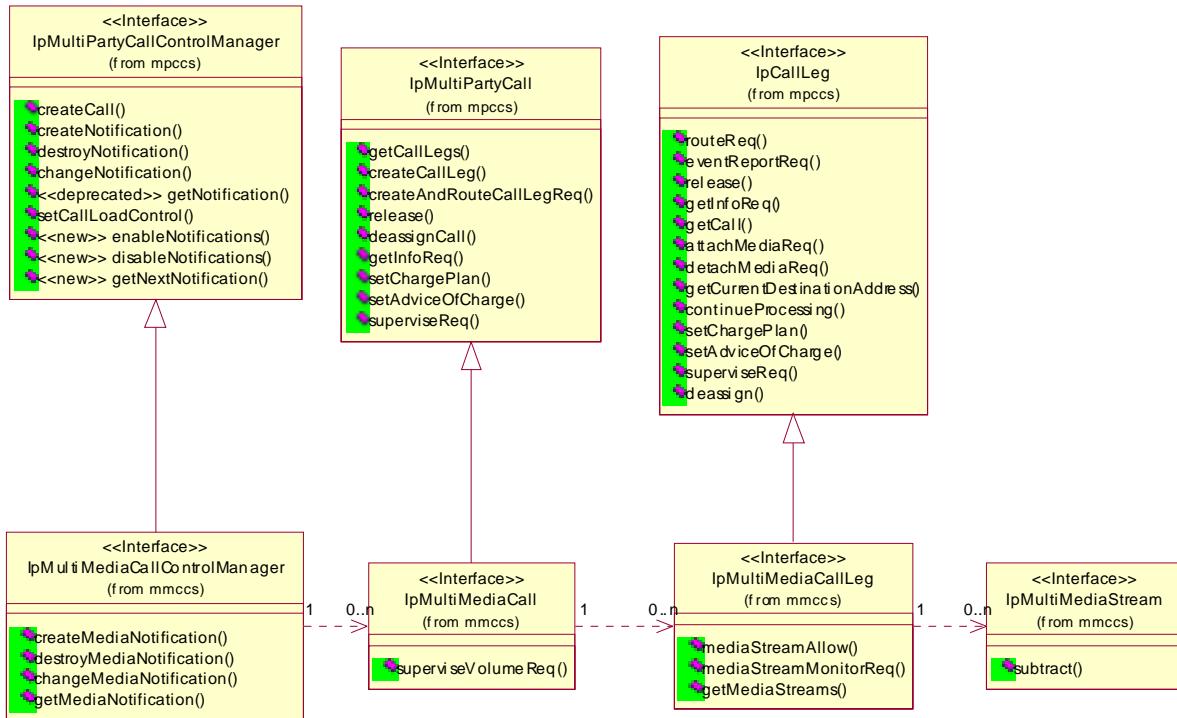
18: The UIcall is released, since no further announcements are needed.

19: In case the user answers that the call should continue, the supervision is reset with a new maximum number of allowed bytes. (note that this might have charging consequences, not shown)

20: If the user answered that the call should not continue, the call is released.

## 5 Class Diagrams



**Figure: Application Interfaces****Figure: Service Interfaces**

## 6 MultiMedia Call Control Service Interface Classes

The MultiMedia Call Control service enhances the functionality of the MultiParty Call Control Service with multi-media capabilities.

The MultiMedia Call Control Service is represented by the IpMultiMediaCallControlManager, IpMultiMediaCall, IpMultiMediaCallLeg and IpMultiMediaStream interfaces that interface to services provided by the network. Some methods are asynchronous, in that they do not lock a thread into waiting whilst a transaction performs. In this way, the client machine can handle many more calls, than one that uses synchronous message calls. To handle responses and reports, the developer must implement IpAppMultiMediaCallControlManager, IpAppMultiMediaCall and IpAppMultiMediaCallLeg to provide the callback mechanism.

To handle the multi-media aspects of a call the concept of media stream is introduced. A media stream is bi-directional media stream and is associated with a call leg. These media streams are usually negotiated between the terminals in the call. The multi-party Call Service gives the application control over the media streams associated with the legs in a multi-media call in the following way:

- the application can be triggered on the establishment of a media stream that meets the application defined characteristics.
- the application can monitor on the establishment (addition) or release (subtraction) of media streams of an ongoing call.
- the application can allow or deny the establishment of media streams (provided the stream establishment was monitored/notified in interrupt mode).
- the application can explicitly subtract already established media streams.
- the application can request the media streams associated with a specific leg.

## 6.1 Interface Class IpMultiMediaCallControlManager

Inherits from: IpMultiPartyCallControlManager

The Multi Media Call Control Manager is the factory interface for creating multimedia calls. The multi-media call control manager interface provides the management functions to the multi-media call control service. The application programmer can use this interface to create, destroy, change and get media stream related notifications.

<pre> &lt;&lt;Interface&gt;&gt; IpMultiMediaCallControlManager </pre>
<pre> createMediaNotification (applInterface : in IpAppMultiMediaCallControlManagerRef,     notificationMediaRequest : in TpNotificationMediaRequest) : TpAssignmentID destroyMediaNotification (assignmentID : in TpAssignmentID) : void changeMediaNotification (assignmentID : in TpAssignmentID, notificationMediaRequest : in     TpNotificationMediaRequest) : void getMediaNotification () : TpMediaNotificationRequestedSet </pre>

### 6.1.1 Method createMediaNotification()

This method is used to create media stream notifications so that events can be sent to the application.

This applies both to callsetup media (e.g., SIP initial INVITE or H.323 with faststart) and for media setup during the call.

This is the first step an application has to do to get initial notifications of media streams happening in the network. When such an event happens, the application will be informed by reportMediaNotification(). In case the application is interested in other events during the context of a particular call session it has to use the mediaStreamMonitorReq() method on the Multi-Media call leg object.

The createMediaNotification method is purely intended for applications to indicate their interest to be notified when certain media stream events take place. It is possible to subscribe to a certain media stream event for a whole range of addresses, e.g. the application can indicate it wishes to be informed when a call is made to any number starting with 800.

If some application already requested notifications with criteria that overlap the specified criteria, the request is refused with P\_INVALID\_CRITERIA. The criteria are said to overlap if both originating and terminating ranges overlap and the same number plan is used.

If the same application requests two notifications with exactly the same criteria but different callback references, the second callback will be treated as an additional callback. Both notifications will share the same assignmentID. The gateway will always use the most recent callback. In case this most recent callback fails the second most recent is used. In case the createMediaNotification contains no callback, at the moment the application needs to be informed the gateway will use as callback the one that has been registered by setCallback().

Returns assignmentID: Specifies the ID assigned by the multi-media call control manager interface for this newly-created notification.

*Parameters***appInterface : in IpAppMultiMediaCallControlManagerRef**

Specifies a reference to the application interface, which is used for callbacks.

**notificationMediaRequest : in TpNotificationMediaRequest**

The mediaMonitorMode is a parameter of TpMediaStreamRequest and can be in interrupt or in notify mode. If in interrupt mode the application has to specify which media streams are allowed by calling mediaStreamAllow on the callLeg.

The notificationMediaRequest parameter specifies the event specific criteria used by the application to define the event required. This is the media portion of the criteria. Only events that meet the notificationMediaRequest are reported.

Individual addresses or address ranges may be specified for the destination and/or origination.

*Returns***TpAssignmentID***Raises*

**TpCommonExceptions, P\_INVALID\_CRITERIA, P\_INVALID\_INTERFACE\_TYPE, P\_INVALID\_EVENT\_TYPE**

### 6.1.2 Method destroyMediaNotification()

This method is used by the application to disable Multi Media Channel notifications

*Parameters***assignmentID : in TpAssignmentID**

Specifies the assignment ID given by the Multi Media call control manager interface when the previous enableMediaNotification was called. If the assignment ID does not correspond to one of the valid assignment IDs, the exception P\_INVALID\_ASSIGNMENTID will be raised.

*Raises***TpCommonExceptions**

### 6.1.3 Method changeMediaNotification()

This method is used by the application to change the event criteria introduced with createMediaNotification. Any stored criteria associated with the specified assignmentID will be replaced with the specified criteria.

*Parameters***assignmentID : in TpAssignmentID**

Specifies the ID assigned by the multi-media call control manager interface for the media stream notification. If two callbacks have been registered under this assignment ID both of them will be disabled.

**notificationMediaRequest : in TpNotificationMediaRequest**

Specifies the new set of event specific criteria used by the application to define the event required. Only events that meet these criteria are reported.

*Raises*

**TpCommonExceptions, P\_INVALID\_ASSIGNMENT\_ID, P\_INVALID\_CRITERIA, P\_INVALID\_EVENT\_TYPE**

### 6.1.4 Method getMediaNotification()

This method is used by the application to query the event criteria set with createMediaNotification or changeMediaNotification.

Returns notificationsMediaRequested: Specifies the notifications that have been requested by the application.

*Parameters*

No Parameters were identified for this method

*Returns*

**TpMediaNotificationRequestedSet**

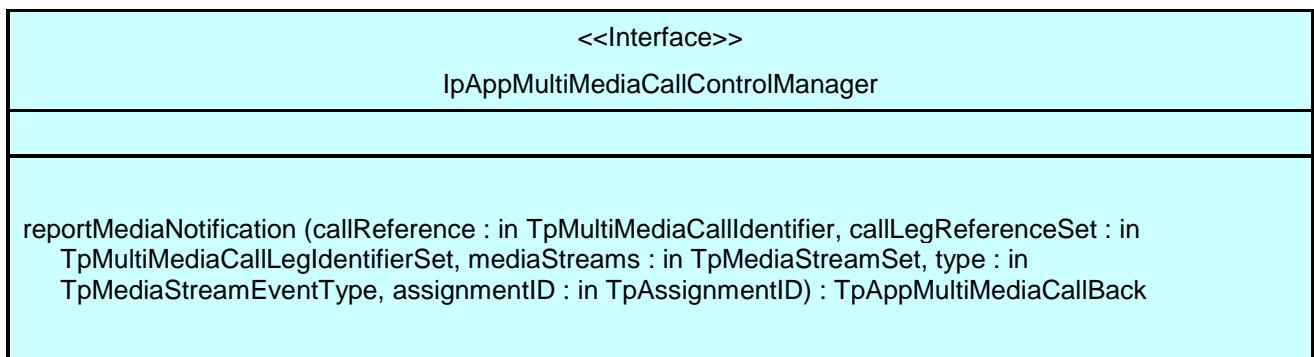
*Raises*

**TpCommonExceptions**

## 6.2 Interface Class IpAppMultiMediaCallControlManager

Inherits from: IpAppMultiPartyCallControlManager

The Multi Media call control manager application interface provides the application call control management functions to the multi media call control service.



### 6.2.1 Method reportMediaNotification()

This method is used to inform the application about the establishment of media streams.

If the corresponding monitor was in interrupt mode, then the application has to allow or deny the streams using mediaStreamAllow() method.

Returns appInterface : Specifies a reference to the application interface which implements the callback interface for the new call.

Returns appMultiMediaCallBack: Specifies references to the application interface which implements the callback interface for the new multi-media call and/or new call leg. This parameter may be null if the notification is being given in NOTIFY mode

*Parameters***callReference : in TpMultiMediaCallIdentifier**

Specifies the call interface on which the media streams were added or subtracted. It also gives the corresponding sessionID.

**callLegReferenceSet : in TpMultiMediaCallLegIdentifierSet**

Specifies set of all callLeg references (interface and sessionID) for which the media streams were established or subtracted.

First in the set is the reference to the originating callLeg. It indicates the call leg related to the originating party. In case there is a destination call leg this will be the second leg in the set. from the notificationInfo can be found on whose behalf the notification was sent.

However, this parameter will be null if the notification is being given in NOTIFY mode

**mediaStreams : in TpMediaStreamSet**

Specifies all the media streams that are established. Note that this can be more media streams than requested in the createMediaNotification, e.g., when faststart is used in H.323 or in SIP when an INVITE method with SDP media stream parameters is used.

**type : in TpMediaStreamEventType**

Refers to the type of event on the media stream, i.e., added or subtracted.

**assignmentID : in TpAssignmentID**

Specifies the assignment id which was returned by the createMediaNotification() method. The application can use assignment id to associate events with event specific criteria and to act accordingly.

*Returns***TpAppMultiMediaCallBack**

## 6.3 Interface Class IpMultiMediaCall

Inherits from: IpMultiPartyCall

<<Interface>> <b>IpMultiMediaCall</b>     superviseVolumeReq (callSessionID : in TpSessionID, volume : in TpCallSuperviseVolume, treatment : in TpCallSuperviseTreatment) : void

### 6.3.1 Method superviseVolumeReq()

The application calls this method to supervise a call. The application can set a granted data volume this call.

*Parameters***callSessionID : in TpSessionID**

Specifies the call session ID of the call.

**volume : in TpCallSuperviseVolume**

Specifies the granted time in milliseconds for the connection.

**treatment : in TpCallSuperviseTreatment**

Specifies how the network should react after the granted volume expired.

*Raises***TpCommonExceptions, P\_INVALID\_SESSION\_ID**

## 6.4 Interface Class IpAppMultiMediaCall

Inherits from: IpAppMultiPartyCall

The application multi-media call interface contains the callbacks that will be used from the multi-media call interface for asynchronous results to requests performed by the application. The application should implement this interface.

<pre>&lt;&lt;Interface&gt;&gt; IpAppMultiMediaCall</pre>
<pre>superviseVolumeRes (callSessionID : in TpSessionID, report : in TpCallSuperviseReport, usedVolume : in TpCallSuperviseVolume) : void superviseVolumeErr (callSessionID : in TpSessionID, errorIndication : in TpCallError) : void</pre>

### 6.4.1 Method superviseVolumeRes()

This asynchronous method reports a call supervision event to the application when it has indicated its interest in these kind of events.

It is also called when the connection is terminated before the supervision event occurs. Furthermore, this method is invoked as a response to the request also when a tariff switch happens in the network during an active call.

*Parameters***callSessionID : in TpSessionID**

Specifies the call session ID of the call

**report : in TpCallSuperviseReport**

Specifies the situation which triggered the sending of the call supervision response.

**usedVolume : in TpCallSuperviseVolume**

Specifies the used time for the call supervision (in milliseconds).

## 6.4.2 Method superviseVolumeErr()

This asynchronous method reports a call supervision error to the application.

*Parameters*

**callSessionID : in TpSessionID**

Specifies the call session ID of the call.

**errorIndication : in TpCallError**

Specifies the error which led to the original request failing.

## 6.5 Interface Class IpMultiMediaCallLeg

Inherits from: IpCallLeg

The Multi-Media call leg represents the signalling relationship between the call and an address. Associated with the signalling relationship there can be multiple media channels. Media channels can be started and stopped by the terminals themselves. The application can monitor on these changes and influence them.

<b>&lt;&lt;Interface&gt;&gt;</b> <b>IpMultiMediaCallLeg</b>
mediaStreamAllow (callLegSessionID : in TpSessionID, mediaStreamList : in TpSessionIDSet) : void mediaStreamMonitorReq (callLegSessionID : in TpSessionID, mediaStreamEventCriteria : in TpMediaStreamRequestSet) : void getMediaStreams (callLegSessionID : in TpSessionID) : TpMediaStreamSet

### 6.5.1 Method mediaStreamAllow()

This method can be used to allow setup of a media stream that was reported by a mediaStreamMonitorRes method.

*Parameters*

**callLegSessionID : in TpSessionID**

Specifies the call leg session ID of the call leg.

**mediaStreamList : in TpSessionIDSet**

Refers to the media streams (sessionIDs) as received in the mediaStreamMonitorRes() or in the reportMediaNotification() that is allowed to be established.

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID**

### 6.5.2 Method mediaStreamMonitorReq()

With this method the application can set monitors on the addition and subtraction of media streams. The monitors can either be general or restricted to certain types of codecs.

Monitoring on addition of media streams can be done in either interrupt or notify mode. In the first case the application has to allow or deny the establishment of the stream with mediaStreamAllow.

Monitoring on subtraction of media streams is only allowed in notify mode.

*Parameters*

**callLegSessionID : in TpSessionID**

Specifies the session ID of the call leg.

**mediaStreamEventCriteria : in TpMediaStreamRequestSet**

Specifies the event specific criteria used by the application to define the event required. The mediaMonitorMode .is a parameter of TpMediaStreamRequest and can be in interrupt or in notify mode. If in interrupt mode the application has to respond with mediaStreamAllow().

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID, P\_INVALID\_CRITERIA, P\_INVALID\_EVENT\_TYPE**

### 6.5.3 Method getMediaStreams()

This method is used to return all currently established media streams for the leg.

*Parameters*

**callLegSessionID : in TpSessionID**

This method is used to return all currently open media channels for the leg,

*Returns*

**TpMediaStreamSet**

*Raises*

**TpCommonExceptions, P\_INVALID\_SESSION\_ID**

## 6.6 Interface Class IpAppMultiMediaCallLeg

Inherits from: IpAppCallLeg

The application multi-media call leg interface contains the callbacks that will be called from the multi-media call leg for asynchronous results to requests performed by the application. The application should implement this interface.

<p style="text-align: center;">&lt;&lt;Interface&gt;&gt;</p> <p style="text-align: center;">IpAppMultiMediaCallLeg</p>
<p>mediaStreamMonitorRes (callLegSessionID : in TpSessionID, streams : in TpMediaStreamSet, type : in TpMediaStreamEventType) : void</p>

### 6.6.1 Method mediaStreamMonitorRes()

This method is used to inform the application about the media streams that are being established (added) or subtracted.

If the corresponding request was done in interrupt mode, the application has to allow or deny the media streams using mediaStreamAllow().

*Parameters*

**callLegSessionID : in TpSessionID**

Specifies the session ID of the call leg for which the media channels are opened or closed.

**streams : in TpMediaStreamSet**

Specifies all the media streams that are added. Note that this can be more media streams than requested in the createMediaNotification, e.g., when faststart is used in H.323 or SIP INVITE with SDP media stream parameters is used.

**type : in TpMediaStreamEventType**

Refers to the type of event on the media stream, i.e., added or subtracted.

## 6.7 Interface Class IpMultiMediaStream

Inherits from: IpService

The Multi Media Stream Interface represents a bi-directional information stream associated with a call leg. Currently, the only available method is to subtract the media stream.

<p style="text-align: center;">&lt;&lt;Interface&gt;&gt;</p> <p style="text-align: center;">IpMultiMediaStream</p>
<p>subtract (mediaStreamSessionID : in TpSessionID) : void</p>

### 6.7.1 Method subtract()

This method can be used to subtract the multi-media stream.

*Parameters***mediaStreamSessionID : in TpSessionID**

Specifies the sessionID for the media stream.

*Raises***TpCommonExceptions, P\_INVALID\_SESSION\_ID**

---

## 7 MultiMedia Call Control Service State Transition Diagrams

There are no State Transition Diagrams for the MultiMedia Call Control Service package

## 8 Multi-Media Call Control Data Definitions

This clause provides the Multi-Media call control data definitions necessary to support the API specification.

The general format of a data definition specification is described below.

- Data Type

This shows the name of the data type.

- Description

This describes the data type.

- Tabular Specification

This specifies the data types and values of the data type.

- Example

If relevant, an example is shown to illustrate the data type.

All data types referenced in the present document but not defined in this clause are defined either in the common call control data definitions in 3GPP TS 29.198-4-1 or in the common data definitions which may be found in 3GPP TS 29.198-2.

### 8.1 Event Notification Data Definitions

#### 8.1.1 TpMediaStreamRequestSet

Defines a [Numbered Set of Data Elements](#) of TpMediaStreamRequest

#### 8.1.2 TpMediaStreamRequest

Defines the [Sequence of Data Elements](#) that specify the type of media stream.

Sequence Element Name	Sequence Element Type
Direction	TpMediaStreamDirection
DataTypeRequest	TpMediaStreamDataTypeRequest
MediaMonitorMode	TpCallMonitorMode

#### 8.1.3 TpMediaStreamDirection

Defines the direction in which the media stream is established (as seen from the leg).

Name	Value	Description
P_SEND_ONLY	0	Indicates that the offerer is only willing to send this media stream
P_RECEIVE_ONLY	1	Indicates that the offerer is only willing to receive this media stream
P_SEND_RECEIVE	2	Indicates that the offerer is willing to send and receive this media stream

### 8.1.4 TpMediaStreamDataTypeRequest

Defines the [Tagged Choice of Data Elements](#) that specify the media type and associated codecs that are of interest.

	Tag Element Type	
	TpMediaType	

Tag Element Value	Choice Element Type	Choice Element Name
P_AUDIO	TpAudioCapabilitiesType	Audio
P_VIDEO	TpVideoCapabilitiesType	Video
P_DATA	TpDataCapabilities	Data

### 8.1.5 TpAudioCapabilitiesType

Defines the audio codec. The requested capabilities can be indicated by adding the values together (i.e., a logical OR function). e.g., 28 indicates interest in all G.722 codes (4+8+16).

Name	Value	Description
P_G711_64K	1	g.711 on 64k, both alaw and ulaw
P_G711_56K	2	g.711 on 56k, both alaw and ulaw
P_G722_64K	4	
P_G722_56K	8	
P_G722_48K	16	
P_G7231	32	
P_G728	64	
P_G729	128	
P_G729_ANNEX_A	256	
P_IS1172	512	
P_IS1318	1024	
P_G729_ANNEXB	2048	
P_G729_ANNEXA_AND_B	4096	
P_G7231_ANNEX_C	8192	
P_GSM_FULLRATE	16384	
P_GSM_HALFRATE	32768	
P_GSM_ENHANCED	65536	

### 8.1.6 TpVideoCapabilitiesType

Defines the video codec. The requested capabilities can be indicated by adding the values together (i.e., a logical OR function). e.g., 3 indicates both H.261 and H.262 codecs.

Name	Value	Description
P_H261	1	
P_H262	2	
P_H263	4	
P_IS11172	8	

### 8.1.7 TpDataCapabilities

A TpInt32 defining the minimum maxBitRate in bit/s. I.e., all data media streams whose maxBitRate exceeds this number are reported.

### 8.1.8 TpMediaStreamEventType

Defines the action performed on the media stream.

Name	Value	Description
P_MEDIA_STREAM_ADDED	0	The media stream is added
P_MEDIA_STREAM_SUBTRACTED	1	The media stream is subtracted.

### 8.1.9 TpMediaStreamSet

Defines a [Numbered Set of Data Elements](#) of TpMediaStream

### 8.1.10 TpMediaStream

Defines the [Sequence of Data Elements](#) that specify the type of media stream.

Sequence Element Name	Sequence Element Type
Direction	TpMediaStreamDirection
DataType	TpMediaStreamDataType
ChannelSessionID	TpSessionID
MediaStream	IpMultiMediaStream

### 8.1.11 TpMediaStreamDataType

Defines the type of the reported media stream. It is identical to [TpMediaStreamDataTypeRequest](#), only now the values are not used as a mask, but as the actual codec should be indicated for audio and video. For data the actual maximum bit rate is indicated.

## 8.2 Multi-Media Call Control Data Definitions

### 8.2.1 IpMultiMediaCall

Defines the address of an IpMultiMediaCall Interface.

### 8.2.2 IpMultiMediaCallRef

Defines a [Reference](#) to type IpMultiMediaCall.

### 8.2.3 IpAppMultiMediaCall

Defines the address of an IpAppMultiMediaCall Interface.

### 8.2.4 IpAppMultiMediaCallRef

Defines a [Reference](#) to type IpAppMultiMediaCall.

### 8.2.5 IpMultiMediaCallLeg

Defines the address of an `IpMultiMediaCallLeg` Interface.

### 8.2.6 IpMultiMediaCallLegRef

Defines a [Reference](#) to type `IpMultiMediaCallLeg`.

### 8.2.7 IpAppMultiMediaCallLeg

Defines the address of an `IpAppMultiMediaCallLeg` Interface.

### 8.2.8 IpAppMultiMediaCallLegRef

Defines a [Reference](#) to type `IpAppMultiMediaCallLeg`.

### 8.2.9 TpAppMultiMediaCallLegRefSet

Defines a [Numbered Set of Data Elements](#) of `IpAppMultiMediaCallLegRef`

### 8.2.10 TpMultiMediaCallIdentifier

Defines the Sequence of Data Elements that unambiguously specify the MultiMediaCall object

Sequence Element Name	Sequence Element Type	Sequence Element Description
MMCallReference	<code>IpMultiMediaCallRef</code>	This element specifies the interface reference for the call object.
MMCallSessionID	<code>TpSessionID</code>	This element specifies the call session ID of the call created.

### 8.2.11 TpMultiMediaCallIdentifierSet

Defines a Numbered Set of Data Elements of `TpMultiMediaCallIdentifier`

### 8.2.12 TpMultiMediaCallLegIdentifier

Defines the Sequence of Data Elements that unambiguously specify the Call Leg object

Sequence Element Name	Sequence Element Type	Sequence Element Description
MMCAllLegReference	<code>IpMultiMediaCallLegRef</code>	This element specifies the interface reference for the callLeg object.
MMCAllLegSessionID	<code>TpSessionID</code>	This element specifies the callLeg session ID of the call created.

### 8.2.13 IpAppMultiMediaCallControlManager

Defines the address of an `IpAppMultiMediaCallControlManager` Interface.

### 8.2.14 IpAppMultiMediaCallControlManagerRef

Defines a [Reference](#) to type `IpAppMultiMediaCallControlManager`.

### 8.2.15 TpAppMultiMediaCallBack

Defines the Tagged Choice of Data Elements that references the application callback interfaces

	<b>Tag Element Type</b>	
	TpAppMultiMediaCallBackRefType	

<b>Tag Element Value</b>	<b>Choice Element Type</b>	<b>Choice Element Name</b>
P_APP_CALLBACK_UNDEFINED	NULL	Undefined
P_APP_MULTIMEDIA_CALL_CALLBACK	IpAppMultiMediaCallRef	AppMultiMediaCall
P_APP_CALL_LEG_CALLBACK	IpAppMultiMediaCallLegRef	AppMultiMediaCallLeg
P_APP_CALL_AND_CALL_LEG_CALLBACK	TpAppMultiMediaCallLegCallBack	AppMultiMediaCallAndCallLeg

### 8.2.16 TpAppMultiMediaCallBackRefType

Defines the type application call back interface.

<b>Name</b>	<b>Value</b>	<b>Description</b>
P_APP_CALLBACK_UNDEFINED	0	Application Call back interface undefined
P_APP_MULTIMEDIA_CALL_CALLBACK	1	Application Multi-Media Call interface referenced
P_APP_CALL_LEG_CALLBACK	2	Application Multi-Media CallLeg interface referenced
P_APP_CALL_AND_CALL_LEG_CALLBACK	3	Application Multi-Media Call and CallLeg interface referenced

### 8.2.17 TpAppMultiMediaCallLegCallBack

Defines the Sequence of Data Elements that references a call and a call leg application interface.

<b>Sequence Element Name</b>	<b>Sequence Element Type</b>	
AppMultiMediaCall	IpAppMultiMediaCallRef	
AppCallLegSet	TpAppMultiMediaCallLegRefSet	Specifies the set of all call leg call back references. First in the set is the reference to the call back of the originating callLeg. In case there is a call back to a destination call leg this will be second in the set.

### 8.2.18 TpCallSuperviseVolume

Defines the Sequence of Data Elements that specify the amount of volume that is allowed to be transmitted for the specific connection.

<b>Sequence Element Name</b>	<b>Sequence Element Type</b>	<b>Sequence Element Description</b>
VolumeQuantity	TpInt32	This data type is identical to a TpInt32, and defines the quantity of the granted volume that can be transmitted for the specific connection.
VolumeUnit	TpInt32	This data type is identical to a TpInt32, and defines the unit of the granted volume that can be transmitted for the specific connection. Unit must be specified as $10^n$ number of bytes, where n denotes the power. When the unit is for example in kilobytes, VolumeUnit must be set to 3.

### 8.2.19 TpNotificationMediaRequest

Defines the Sequence of Data Elements that specify the criteria for a media stream notification

<b>Sequence Element Name</b>	<b>Sequence Element Type</b>	<b>Description</b>
MediaNotificationScope	TpCallNotificationScope	Defines the scope of the notification request.
MediaStreamsRequested	TpMediaStreamRequestSet	Defines the media stream events which are requested

### 8.2.20 TpMediaNotificationRequested

Defines the Sequence of Data Elements that specify the criteria relating to event requests.

<b>Sequence Element Name</b>	<b>Sequence Element Type</b>
AppNotificationMediaRequest	TpNotificationMediaRequest
AssignmentID	TpInt32

### 8.2.21 TpMediaNotificationsRequestedSet

Defines a numbered Set of Data Elements of TpMediaNotificationRequested

---

## Annex A (normative): OMG IDL Description of Multi-Media Call Control SCF

The OMG IDL representation of this interface specification is contained in the text file mmccs.idl (contained in archive 291980404IDL.ZIP) which accompany the present document.

---

## Annex B (informative): W3C WSDL Description of Multi-Media Call Control SCF

The W3C WSDL representation of this specification is contained in text files (mmccs.wsdl contained in archive 291980404WSDL.ZIP) which accompanies the present document.

---

## Annex C (informative): Java API Description of the Call Control SCFs

The Java API representation of this specification can be obtained from the following URL:

- Java Call Control (<http://jcp.org/jsr/detail/21.jsp>)

Each JSR webpage contains a table identifying the relationships between the different versions of the Parlay, ETSI/OSA, 3GPP/OSA and JAIN SPA specifications. In addition, each JAIN SPA specification version indicates to which Parlay, ETSI/OSA and 3GPP/OSA specification versions it corresponds to.

## Annex D (informative): Change history

Change history							
Date	TSG #	TSG Doc.	CR	Rev	Subject/Comment	Old	New
Mar 2001	CN_11	NP-010134	047	-	CR 29.198: for moving TS 29.198 from R99 to Rel 4 (N5-010158)	3.2.0	1.0.0
June 2001	CN_12	NP-010327	--	--	Approved at TSG CN#12 and placed under Change Control	2.0.0	4.0.0
Sep 2001	CN_13	NP-010467	001	--	Changing references to JAIN	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	002	--	Correction of text descriptions for methods enableCallNotification and createNotification	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	003	--	Specify the behaviour when a call leg times out	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	004	--	Removal of Faulty state in MPCCS Call State Transition Diagram and method callFaultDetected in MPCCS in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	005	--	Missing TpCallAppInfoSet description in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	006	--	Redirecting a call leg vs. creating a call leg clarification in OSA R4	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	007	--	Introduction of MPCC Originating and Terminating Call Leg STDs for IpCallLeg	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	008	--	Corrections to SetChargePlan() Addition of PartyToCharge parameter	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	009	--	Corrections to SetChargePlan()	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	010	--	Remove distinction between final- and intermediate-report	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	011	--	Inclusion of TpMediaType	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	012	--	Corrections to GCC STD	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	013	--	Introduction of sequence diagrams for MPCC services	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	014	--	The use of the REDIRECT event needs to be illustrated	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	015	--	Corrections to SetCallChargePlan()	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	016	--	Add one additional error indication	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	017	--	Corrections to Call Control – GCCS Exception handling	4.0.0	4.1.0
Sep 2001	CN_13	NP-010467	018	--	Corrections to Call Control – Errors in Exceptions	4.0.0	4.1.0
Dec 2001	CN_14	NP-010597	019	--	Replace Out Parameters with Return Types	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	020	--	Removal of time based charging property	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	021	--	Make attachMedia() and detachMedia() asynchronous	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	022	--	Correction of treatment datatype in superviseReq on call leg	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	023	--	Corrections to Call Control Data Types	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	024	--	Correction to Call Control (CC)	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	025	--	Amend the Generic Call Control introductory part	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	026	--	Correction in TpCallEventType	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	027	--	Addition of missing description of RouteErr()	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	028	--	Misleading description of createAndRouteCallLegErr()	4.1.0	4.2.0
Dec 2001	CN_14	NP-010597	029	--	Correction to values of TpCallNotificationType, TpCallLoadControlMechanismType	4.1.0	4.2.0
Dec 2001	CN_14	NP-010695	030	--	Correction of method getLastRedirectionAddress	4.1.0	4.2.0
Mar 2002	CN_15	NP-020106	031	--	Add P_INVALID_INTERFACE_TYPE exception to IpService.setCallback() and IpService.setCallbackWithSessionID()	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	032	--	Correction of Event Subscription/Notification Data Type	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	033	--	Correction of parameter name in IpCallLeg.routeReq() and in IpCallLeg.setAdviceOfCharge()	4.2.0	4.3.0
Mar 2002	CN_15	NP-020106	034	--	Clarification of ambiguous Event handling rules	4.2.0	4.3.0
Jun 2002	CN_16	NP-020180	035	--	Correction to TpCallChargePlan	4.3.0	4.4.0
Jun 2002	CN_16	NP-020180	036	--	Correction to CAMEL Service Property values	4.3.0	4.4.0
Jun 2002	CN_16	NP-020181	037	--	Addition of support for Java API technology realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020182	038	--	Addition of support for WSDL realisation	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	039	--	Addition of support for Emergency Telecommunications Service	4.4.0	5.0.0
Jun 2002	CN_16	NP-020183	040	--	Addition of support for Network Controlled Notifications MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	041	--	Changes to getNotification()	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	042	--	Addition of P_UNSUPPORTED_MEDIA release cause to TpReleaseCause	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	043	--	Addition of CAMEL Phase 4 Service Property values	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	044	--	Addition of indication whether SCS supports initially multiple routeReqs in parallel	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	045	--	Explicit exception for continueProcessing when not in interrupted mode	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	046	--	Indication needed that supervision will be ended when call or callLeg is reassigned	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	047	--	Clarify ambiguous Supervision duration	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	048	--	Detach/Attach request illegal during pending Attach/Detach request	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	049	--	Correction of Multi-Party Call Control properties	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	050	--	Correcting the sequence diagram descriptions in GCC and MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	051	--	Correcting erroneous description of UI behaviour in call control	4.4.0	5.0.0

Jun 2002	CN_16	NP-020187	052	--	Correcting the descriptions of sequence diagrams that don't match the diagram	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	053	--	Correcting erroneous references to GCC in MPCC	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	054	--	Addition of the Multi-media APIs to Call control SCF (29.198-4)	4.4.0	5.0.0
Jun 2002	CN_16	NP-020187	055	--	Updating Clause 4 for Release 5	4.4.0	5.0.0
Jun 2002	CN_16	NP-020188	056	--	Splitting of 29.198-04 into 4 separate TSs (sub-parts)	4.4.0	5.0.0
Sep 2002	CN_17	NP-020395	001	--	29.198-04-4 Add text to clarify relationship between 3GPP and ETSI/Parlay OSA specifications	5.0.0	5.1.0

---

## History

<b>Document history</b>		
V5.0.0	June 2002	Publication
V5.1.0	September 2002	Publication